Database Models for Computational Lexicography

Branimir Boguraev-Ted Briscoe-John Carroll-Ann Copestake

1. Introduction

This paper[1] examines, from the perspective of computational lexicography/lexicology, the range of options for representation and storage of lexical data. The discussion is limited to the particular context of extraction of such data, insofar as it is of utility to natural language processing, from machine-readable dictionary sources. We thus ignore, for instance, considerations of building on-line dictionaries for human use, or designing software systems for compiling dictionary entries for inclusion in human dictionaries. Instead, we concentrate on the predominant paradigm for acquisition of lexical information and look at the suitability of existing data models for generic tasks like identification, discovery, extraction and representation of lexical properties of words on the basis of studying large dictionaries available in electronic form.

A growing number of projects in computational lexicography and lexicology use machine-readable forms of published dictionaries (MRDs) as their starting point. Individual goals may vary substantially; they might include, for instance, linguistic analysis of dictionary definitions, statistically-based search for regular patterns across a dictionary source, or semi-automatic derivation of computational lexicons from existing dictionaries. However, a common characteristic of such projects is their shared requirement for a framework which facilitates the mapping from source form (typically a typesetting tape) to a lexical database (LDB).

There are at least three aspects to such frameworks, falling in the general categories of *transduction*, *representation*, and *query*. Outside of the specifics of any particular approach, the ultimate use of a machine-readable dictionary is for extraction of (fragments of) lexical entries. Access to the lexical content of an on-line dictionary is typically provided by a query mechanism; the expressive power of the query language determines the range and granularity of lexical properties extractable from the source. To a certain extent, granularity also depends on the representational scheme employed by the on-line dictionary model. More importantly, however, the nature of dictionary representation crucially constrains the kinds of observations that can be made regarding both implicitly encoded information in the dictionary and the linguistic generalisations reflected in it. Finally, independent of the representation itself, some kind of computational support is required for transducing the raw tape format into a set of dictionary entry records and fields.

These issues are orthogonal; in particular, the choice of mechanism for parsing the dictionary source can be largely independent of the nature of representation adopted for holding dictionary data. Still, work in this area tends to blur the line between parsing and loading; consequently there is no principled distinction between the processes involved in source transduction and those underlying the functionality required from an on-line dictionary. The development of a general purpose appara-

1. Paper presented at *Fourth International Congress on Lexicography (Euralex-VOX)*, Malaga, Spain, August 1990.

tus for entry parsing is emerging as one of the concerns of computational lexicography (see e.g. Neff and Boguraev, 1989, 1991); likewise, proposals are being developed; for a general-purpose, application-independent representational scheme for on-line dictionaries (Sperberg-McQueen and Burnard, 1990). However, there is no consensus yet on what would constitute a general computational model of a dictionary, especially from the viewpoint of what kinds of processing such a model ought to support.

This paper attempts to separate the issue of representation from concerns of parsing and access, and relate it to some of the methodologies for lexical data acquisition. To this end, we present a comparative analysis of the range of existing lexical database formats and examine the contributions, and suitability, of conventional database technology to the design of dictionary databases. Then we suggest a set of criteria for choosing among existing designs, given specific project requirements and constraints. Finally, we argue for a new computational model of the lexicon, which looks forward to applications fully exploiting the power of computers both in the use and compilation of lexical data.

2. Dictionary database models

We identify four classes of dictionary models. These can be divided into *relational* and *hierarchical* formats, with further subdivision into physical, logical, and lexical conceptual hierarchies functioning as the organising principle of the on-line dictionary representation.

2.1. Dictionaries as relational databases

Following a well established notion in database technology, the relational model of a dictionary maps a dictionary entry onto a set of tables. There is no canonical general purpose mapping between lexical attributes and properties, typically found in a dictionary, and entities and relationships, modelled by a relational database. Nonetheless, there is usually a straightforward way of designing a database schema, capable of holding the information visible in a dictionary entry. For this reason, and given the convenience of an established and widely available technology, a number of projects have adopted relational databases as the representational device for their dictionary sources.

As an example, consider the database schema developed for one on-line model of the *Longman Dictionary of Contemporary English* (Procter, 1978; henceforth LDOCE). For the purposes of extracting lexical relations like **is_a, part_of, group_of, degree,** and so forth, Nakamura and Nagao (1988) have focused on few specific fields in LDOCE entries, of which the definition, the example sentences, and the subject code[2] are most prominent. Consequently, the database schema is defined as follows:

2. The Longman tape source contains certain markings, e.g. subject (i.e. semantic domain) code, which are not incorporated in the printed form of the dictionary. This in no way affects the argument in this paper.

headword	**pos**	**def_no**	**g_code**	**b_code**
headword	**pos**	**def_no**	**definition**	
headword	**pos**	**def_no**	**sample_sentence**	

Several observations can be made concerning the relationship between source dictionary entries and their representation in relational format.

Most noticeably, not all of the lexical content in the dictionary has been carried over. While this is not a requirement of the mapping process *per se*, it is characteristic of many relational realisations. Partly, this reflects a feature common to the projects which have adopted relational representation: they typically are interested only in certain aspects of the data available in a dictionary entry, without much concern for more complex relationships between its individual fields and fragments. This is also, however, a consequence from the inherent conflict between the hierarchical organisation of a dictionary entry and the expressive power of the relational model. In order to represent more than one value for an attribute —a common situation in entries predominantly organised as clusters of one-to-many mappings: several senses per word, several definitions per sense, several examples per definition— it is necessary to largely duplicate database records which would differ in a small number of their fields. The desire to avoid reduplication of data, as well as the growing complexity of the schema (and actual database) if («more») fine-grained analysis of the dictionary is required, make the relational model suitable only for projects where the lexical information sought is localised in a small number of source fields, preferably found towards the top (or root) of the conceptual hierarchy underlying the dictionary entry organisation.

A more subtle problem with the relational model is that, once the mapping between entry format and database schema has been cast, certain interesting, and systematic, relationships between aspects of lexical data tend to get lost. This is also a consequence of the limited granularity of representation: once the source analysis has been carried out, certain repeated patterns in entry configurations may be obscured in the breakdown of complete entries into a set of disjoint records.

For instance, during a discussion of certain features of the lexical taxonomy derived from the relational model of LDOCE illustrated above, Nakamura and Nagao (1988), address the problem of tangledness of lexical hierarchies. The issue is to identify and capture dual genus terms, as in the definitions below:

> **dwarf** a person, animal, or plant of much less than the usual size
>
> **hunter** a person or animal that hunts ...

The approach is to detect certain patterns (such as a disjunction) in the definitions and process them as appropriate. This appears to miss out on a different definitional pattern, which also introduces multiple inheritance: the Longman convention of having subdefinitions within a definition is not addressed in the database schema. Consider the following examples (pointers to superordinate concepts, indicated by un-

derscoring, are derived from looking at the subdefinitions within a single word sense —i.e. in a single field of the database):

> **breakout** ... **a** a military attack to break from being surrounded **b** an escape from prison ...
>
> **dictatorship** ... **a** the position or power of a DICTATOR ... **b** the period during which a DICTATOR rules a country

What is important to note here is not that the model fails to capture certain regularity of the source, but that capturing it would require a deeper level of analysis. Indeed, in a different project, Fontenelle and Vanandroye (1989) design a database schema which reflects just this property of LDOCE. They address the problem of finding ergative verbs, and one of the heuristics built into some of the queries they design for this purpose relies on examining verb sense subdefinitions: «Our version of LDOCE is organised in such a way that it is possible to query all word senses of verbs that have *cause* in the definition field, along with letter **b** in the 1-character definition letter field (the latter condition specifies that the word sense has been split into at least two sub-definitions, [but does not necessarily imply that *cause* must be present in the second sub-definition]).»

This retrieves entries like

> **reverse**[2] **1**[T1;I0] **a** to cause (a vehicle) to go backwards **a** of (a vehicle) to go backwards ...

and represents an attempt to 'cast the net wider' by applying a heuristic which relies on looking for causativity in a verb sense definition.

However, the methods described in Fontenelle and Vanandroye (1989) still fail to retrieve a class of verbs which would be exemplified by the following entries:

> **trim** ... **4 a** to move (a sail) into the desired position **b** (of a sail) to move into the desired position ...
>
> **shunt 1 a** to turn (a railway train or carriage) from one track to another ... **b** (of a train) to be turned in this way
>
> **trim 5 a** to arrange the load of (a ship or aircraft) so as to give the desired balance in the water or air **b** (of a ship or aircraft) to balance in this way

The problem here is not in the representation itself; rather, the set of heuristics designed for the task apparently is not comprehensive enough to react to a certain pattern in the definitions. However, as we argue below, it is due to the limitations of granularity and distributed nature of the relational representation that a relevant systematicity has been missed. In other words, the relational model of a dictionary is not especially conducive to browsing, especially in a highly opportunistic mode. We discuss this claim in detail later.

2.2. Dictionaries and hierarchies

The fact that dictionary entries can quite naturally be regarded as shallow hierarchies with an open-ended number of attributes at each level (e.g. word sense clusters within an entry or examples within a definition), suggests that some kind of hierarchical representation would be a more natural match for the data. It turns out, however, that there are different perspectives on the information typically contained in a dictionary entry; consequently, different notions of hierarchy underlie three general classes of hierarchical dictionary models.

The obvious distinction which can be made concerning dictionary entries is that between form and content. Typically, an intricate complex of lexicographic conventions is implemented by an elaborate system of typographic codes and special characters. The visual make-up of an entry does not exist in isolation: different fonts convey different types of lexical data, respectively. The interpretation of those codes, however (in the case, for instance, of parsing a raw type-setting tape), is only possible if the parser is aware of global context. Still, the *presentation* of an entry, in its printed form, is achieved by what is, in effect, a hierarchically structured 'language' for controlling a type-setting device. On the other hand, it is primarily through this same system of lexicographic conventions (and, hence, typesetter control codes) that the hierarchical *organisation* of dictionary entry content is conveyed to the reader.

Depending on whether a database model of a dictionary chooses to highlight the physical, or the logical, structure of its source, we can distinguish between *physical* and *logical* hierarchical representations. In addition, a 'hybrid' model attempts to reconcile some of the conflicts between attempting to maintain both form and content in the database. We will refer to this as *conceptual* hierarchy, in recognition of the fact that, as we discuss below, the primary organisational unit in the database is that of a structured lattice of linguistic concepts, underlying the combined form and content of the dictionary being modelled.

Logical hierarchy

This model of the dictionary offers advantages over the relational one in several respects. It suits lexical intuitions. It avoids the (largely unnecessary) duplication of information concommitant with increased granularity of analysis; consequently, it allows arbitrary depth of analysis. Most importantly, it underlies a structured representation designed to transcribe the majority of existing conventions and notations for writing dictionary entries.

At a certain level of abstraction, lexicographic conventions are expressed by manipulating segments of dictionary entries according to certain rules. An entry embodies, in its printed form, the application of these rules over the sum total of lexical information associated with this word. The rules are designed to achieve as compact presentation as possible. Analysing an entry, in terms of recovering *all* of the data in it, implies also 'undoing' the effect of those compaction procedures. Neff and Boguraev (1989, 1991) discuss in detail the architecture of a dictionary entry parsing system, designed to meet precisely this requirement. Here we present, briefly, the major categories of entry configurations, which the logical hierarchy can naturally accommodate within its representational framework.

Compaction. Segments in dictionary entries occasionally serve more than one function. Consider the entry for «accordion» below, where the string «KEY» acts simultaneously as a part of the definition text, a parenthetical expression, and an implicit cross-reference to another entry in the dictionary:

> **ac.cor.di.on** /.../ *n* a musical instrument that may be carried and whose music is made by pressing the middle part together and so causing air to pass through holes opened and closed by instruments (KEY*s*[1] (2)) worked by the fingers — compare CONCERTINA[1] — see picture at KEYBOARD[1]

A different example is the common convention of having the headword of an entry denote both its print form and hyphenation points, as in **ac.cor.di.on**.

In an unconstrained hierarchical representation, such multi-functional segments can be represented as suitably labelled sister nodes, as the following database fragment illustrates:

```
........
|
+-defn
  |
  +-implicit_xref
  | +-to:          key
  | +-x_morph:     s
  | +-hom_no:      1
  | +-s_no:        2
  |
  +-par_string:    keys.
  +-def_string:    a musical instrument that may be carried and
                   whose music is made by pressing the middle
                   part together and so causing air to pass through
                   holes opened and closed by instruments (keys)
                   worked by the fingers
```

Elision. Often alternative descriptions logically belonging to a given level in the lexical description of a word are compacted by using devices like elision of fragments and (backward and/or forward) scoping of lexical descriptors. As an example, consider the following entry fragments, taken from the *Collins English-German Dictionary*. The explanation for «bagpiper», «Dudelsackpfeifer *or* -bläser *m*», really stands for «(Dudelsackpfeifer *m*) *or* (Dudelsackbläser *m*)»; similar unravelling process should be applied to the segment «**abutment**» *n, (Archit)* Flügel- *or* Wangenmauer *f*» in the entry for «abutment». Without going into details, we note that the LDOCE system of grammar coding uses exactly the same devices for compacting descriptions of complementation patterns and syntactic environments of words.

Embedding. Sometimes dictionaries choose to explain a word in the course of defining another related word by arbitrarily inserting mini-entries in their definitions:

lach.ry.mal /... / *adj* [Wa5] of or concerning tears of the organ (**lachrymal gland**/'... ./) of the body that produces them

Recovery of what is, logically, a separate entry requires reasoning over the tree shape from the perspective of the specific position (and scoping) of the embedded segment.

Context sensitivity. This property of dictionaries reflects another pervasive convention: units which appear identical from a typographic point of view (like, for instance, everything typeset in secondary bold) may perform different functions depending on local and/or global context. More importantly—and less obvious—the notational conventions for a logical unit within an entry may persist across different contexts, and the representation should reflect this. Consider this entry:

book[1] / ... / *n* **1** a collection of sheets of paper fastened together as a thing to be read, or to be written in ... **2** one of the main divisions or parts of a larger written work (as of a long poem or the Bible) **3** the words of a light musical play: *Oscar Hammerstein II wrote the book of "Oklahoma", and Richard Rodgers wrote the music* -- compare LIBRETTO ... **12 throw the book at (someone)** (esp. of the police or a judge) to make all possible charges against (someone) — see also BOOKS

On the face of it, the change of typeface to small capitals in the segment «LIBRETTO» indicates the use of a word outside of the controlled core vocabulary (Procter, 1978). This is the minimal analysis which might be assigned to the particular font controlled character, and carried over to the dictionary representation. However, the typographical convention here is used to signal an (implicit) cross-reference to another entry. In terms of representation, we are faced with several possibilities. Discarding the 'noisy' typesetter control might result in a data structure which represents the fact that «libretto» is an (explicit) cross-reference of the straightforward *see* category (as opposed to, for instance, *compare, opposite,* or *see picture at*). Alternatively, we might follow the minimal analysis and retain a trace of the font change:

```
...[begin[small_caps]]libretto[end[small_caps]]...
```

Both of these representations are clearly impoverished; of special interest here, however, is the fact that an alternative

```
... implicit_xref = "libretto"
```

is equally lacking: it fails to capture the fact that the string in question is an implicit cross-reference within an explicit cross-reference applying to the third sense definition of the first homograph for «book».

This is precisely the knowledge that a hierarchical model represents naturally, by encoding it in a path from the root of the entry tree to this particular terminal node:

```
LDOCE:
  entry
    .homograph
       .sense_def
          .explicit_xref
             .implicit_xref
                .to: "libretto" ;
```

It is this notion of context-driven decoding of a simple font change code, such as **small_caps**, that assigns non-atomic 'labels' (i.e. composite paths) to text fragments within an entry. Having functional properties of lexical segments defined decompositionally is important, since now we have a richer language for describing them; this goes beyond an interpretation of the immediate tag (terminal label) they carry, by considering their complete or partial path indicating their overall participation in the lexical make-up of a language.

It turns out that the path concept can be interpreted in a way which identifies it with the notion of a semantic field in traditional linguistics and lexical semantics. As a result of this, and given a suitable access mechanism, the 'logical hierarchy' model of a dictionary becomes a particularly powerful vehicle for identification and extraction of a variety of lexical data. As an example, as well as to illustrate an access mechanism designed to exploit structured paths, the following query constructs a list of noun pairs:

```
entry                          +-------- CONDITION ----+
+-hdw: _word                   | _ixref  ne  ""        |
|                              |                       |
+-superhom                     |                       |
  |                            |                       |
  +-syncat: "*"                |                       |
  |                            |                       |
  +-sense_def                  +-----------------------+
    +-aux_def
      +-implicit_xref
        +-to: _ixref

+------------------------------------ FORMAT -----+
|   {\bf _word}                  {\it _ixref}      |
|                                                  |
+--------------------------------------------------+
  LDB: ldoce ldb *      ANSWER:            OUTPUT:

 hermetic                          airtight
 hyperbole                         exaggeration
 keep back                         withhold
 lead                              clue
 impoverish                        deplete
 meddle                            interfere
```

It is interesting to note that a query stated in purely structural terms—list the headword and an associated implicit cross-reference within an explicit cross-reference in the entry—maps onto a particular semantic property, in this case synonymy. This is discussed in detail in Boguraev (1991); in section 3 below we compare the different models from this perspective.

Physical hierarchy

While particularly well suited to most of the tasks of computational lexicology (see e.g. Boguraev, 1991), modelling a dictionary as a logical hierarchy fails to meet at least one crucial requirement of computational lexicography. In particular, it offers no natural way of supporting an inverse transformation to that of parsing a dictionary source: the model is designed to encode complex structural relationships between fields and contents of a dictionary entry, but it cannot aid the derivation of a visual equivalent of these relationships (conventionally denoted by intricate typography). Once the dictionary has been analysed and converted to a database, the only way to achieve a mapping back to its original printed form is by special-purpose programs tuned to a particular transformation. The compaction, elision, and other rules which implement lexicographic conventions will have to be 'hard-wired' into these programs. Even so, in many cases the transformations are ambiguous, and there is no guarantee that the reconstructed source would be identical in presentation to the original. In essence, in the course of highlighting the logical (or content-full) aspects of the dictionary, some of its physical characteristics get irretrievably lost.

The physical characteristics of a dictionary, relating its visual organisation to its internal representation, are defined by the mapping between typesetting control codes and entry format. At least from the perspective of the concerns of this paper, it is important that this mapping be bi-directional, rather than only from raw source to a markup format: an essential prerequisite for being able to mediate between raw text, markup format and visual display is that there be unique path in both directions. In particular, the physical structure of text should support easy and unambiguous recovery of its original appearance in printed form. One of the essential features of dictionary entry organisation—their hierarchical structure—thus migrates to this dictionary model too, even though the emphasis now is on the physical, rather than logical, aspects.

Consequently, and specifically for the purpose of incorporating a model of the dictionary into a (generic) lexicographer's workstation, a number of proposals elaborate the notion of a *tagged* dictionary representation (e.g. Amsler and Tompa, 1988). The most representative work in this paradigm, especially where dictionary analysis is concerned, comes from research at the University of Waterloo on analysing several Oxford University Press dictionaries, including the *Oxford English Dictionary* (Kazman, 1986), and more recently, the *Oxford Advanced Learner's Dictionary of Current English* (OALD Electronic, 1988). The example below, taken from the latter source, illustrates the kinds of tag assignments that tend to be derived from a typesetting tape:

torment /'tɔ:ment/ *n* [C,U] (sthing that causes) severe bodily or mental pain or suffering: *be in ~; suffer ~(s) from an aching tooth; the ~s of jealousy. What a little ~ that child is!* (because it worries, asks constant questions, etc.) ◇ *vt* /tɔ:'ment/ [VP6A,15A] cause severe suffering to; annoy: *~ed with neuralgia / hunger / mosquitoes. Stop tormenting your father by asking silly questions.* **tor.men.tor** /tɔ:'mentə(r)/ *n* sb or sth that ~s

```
<ent h=torment><hwd>tor|ment</hwd><pr><ph>"tO;ment</ph></pr>
<hps ps=n cu=C,U><hsn><def>(sth that causes) severe bodily
or mental pain or suffering</def><ex>be in <hr></ex><ex>suffer
<hr>(s) from an aching tooth</ex><ex>the <hr>s of jealousy</ex>
<ex>what a little <hr> that child is!</ex><def>(because it
worries, asks constant questions, etc)</def></hsn></hps>
<hps ps=vt vp=6A,15A><pr><ph>tO;"ment</ph></pr><hsn><def>
cause severe suffering to</def><def>annoy</def><ex><hr>ed with
neuralgia / hunger / mosquitoes</ex><ex> stop <hr>ing your father
by asking silly questions</ex><cd><cp>tor|men|tor</cp><pr><ph>
tO:"ment%(r)</ph></pr><cps ps=n><csn><def>sb or sth that <hr>s
</def></csn></cps></cd></hsn></hps></ent>
```

On the face of it, both the logical and physical hierarchy models appear to use the notion of a tree structure to hold the data stated in a dictionary entry. This, however, does not make them equivalent: the difference stems from the distinctions between functional and presentation aspects of the information represented by the tree.

One way of looking at these is by comparing the processes of *parsing*, as embodied in a system which concerns itself with recovery of logical structure of a dictionary (an example of such a system is the Dictionary Entry Parser, developed by Neff and Boguraev, 1989, 1991), and that of *tagging*, as assumed by a system capable of the assignments just illustrated. Tagging involves, in principle, no more than identification of entry-internal field delimiters, their interpretation in context and markup of individual components by 'begin-end' brackets. It does not, however, extend to recovery of elided information; nor does it imply explicit structure manipulation.

As an example of alternative target representations of a dictionary source, consider the definition fragment of the third sense for the LDOCE entry for «nuisance»:

> **nui.sance** /'nju:səns ‖ 'nu:-/ *n* **1** a person or animal that annoys or causes trouble, PEST: *Don't make a nuisance of yourself sit down and be quiet!* **2** an action or state of affairs which causes trouble, offence, or unpleasantness: *What a nuisance! I've forgotten my ticket* **3 Commit no nuisance** (as a notice in a public place) Do not use this place as **a** a lavatory **b** a TIP[4]

Assuming a notation in the spirit of that used for the parsed OED, where entry components are bracketed by ‹tag› and ‹/tag› to mark their beginning and end respectively, the tagged version of the fragment would be:

```
... <deftext> Do not use this place as <subdef sdlet=a>
a lavatory </subdef> <subdef sdlet=b> a <i_xrf> tip
<ix_sno> 4 </ix_sno> </i_xrf> </subdef> </deftext> ...
```

In contrast, compare this representation with the second sub-definition fragment in a functional representation for «nuisance»:

```
...
|
+-sub_defn
  |
  +-seq_no:  b
  +-defn
    |
    +-implicit_xref
    | |
    | +-to:     tip
    | +-hom_no: 4
    |
    +-def_string:  do not use this place as a tip
```

In the tagged version, the fact that **"tip"**, in addition to being a substring in a (sub-) definition text is also a key to an implicit cross reference, is represented only configurationally, as a particular pattern of nesting of the **subdef** and **i_xrf** tags. Similarly, there is no explicit statement concerning the existence of a definition-initial substring, common to both sub-definitions; this information is also implicit in the collocations of **deftext** and **subdef** tags and the presence of a non-null string between the two. Thus the structural relationships, explicitly labelled and represented in our LDB format, can only be infered on the basis of the 'semantics' of tag names and their specific collocations in the tagged format.

However, for the purpose of identification and extraction of lexical data, based on opportunistic browsing of LDB's and studies of structural relationships in dictionaries, highly structured and functionally orientated representation is more convenient. Firstly, this is because when coupled with a suitable database interface, even complex queries can be constructed easily and rapidly. Secondly, explicitly annotated structures of individual entries facilitate structural analysis of the entire dictionary: this is a crucial prerequisite for any effort of acquisition of lexical semantic information based on the notion of distributed lexical knowledge (as that discussed in Boguraev, 1991).

As a brief example, consider the following scenario. We wish to test, using a machine-readable dictionary, one particular hypothesis about the lexical organisation of verbs: a verb of gesture or sign made with a part of body can take on an extended meaning of the following type via a process of lexical subordination: «to express by means of [V]-ing». In other words, a sentence like *She smiled* can be assigned the interpretation *She expressed her approval by means of smiling* (Levin, forthcoming). As part of our method, we need to retrieve a set of verbs, capable of functioning both transitively and intransitively, whose definitions contain the substring «to express». Using the query mechanism developed for the on-line Oxford dictionaries, retrieval is achieved by filtering sets of entries through a series of constraints: find all verbs marked transitive; within those, find all that are marked intransitive; and within those, select ones which match the substring membership criterion (**x**, **y**, and **z** stand for numbers indicating size of retrieved samples):

```
>> docs ent including ( "vp=" fby.20 "2A" )
   1:  {X} matches

>> docs ent including ( ( "vp=" fby.20 "6A" )
                        within 1 )
   2:  {Y} matches

>> docs def including ( ( "to " fby "express " )
                        within % )
   3:  {Z} matches
```

Without going into details of the notation, two observations hold. First, it is cumbersome and unintuitive: compare the query above with the one below, stated in terms of a functional specification pattern with constraints on terminal nodes:

```
entry                    +-------- CONDITION ------+
|                        |                         |
|                        | sub ( "to express",     |
+-hdw: _word             |       _def ) gt 0       |
|                        | sub ( "v", _pos ) eq 1  |
+-superhom               | _gc1 eq "T1"            |
  |                      | _gc2 eq "I0"            |
  +-syncat: _pos         |                         |
  +-gcode: _gc1          |                         |
  +-gcode: _gc2          |                         |
  |                      |                         |
  +-sense_def            +-------------------------+
    |
    +-def_string: _def

+-------------------------------- FORMAT --------+
|      _word                                     |
|                                                |
+------------------------------------------------+
LDB: ldoce ldb *    ANSWER:           OUTPUT:
```

Incidentally, some of the verbs retrieved by these queries are:

act out, applaud, articulate, babble, boo, bow, cluck, cry, dance, enunciate, gag, giggle, glare, growl, drumble, grunt, hold, howl, keen, kiss, knock, lament, laugh, look, moan, plot, project, protest, purr, registered, roar, scowl, scream, see, signal, slant, slobber, smile, snort, sound, speak, spin, spit, storm, talk, tell, threaten, turn, tut-tut, unload, vote, wave, wish, work, write.

While it is true that the properties of an access mechanism are in principle separate from the underlying representation, ultimately it is the features of this representation that determine the parameters of a query language. This leads us to the second observation: the physical hierarchy model employs representation which is essentially a character stream, with tags spliced into the data itself. A query language over such a representation should provide string calculus primitives. Overall, this makes asking questions in terms of structural (configurational) patterns very difficult—if not impossible. We discuss this in more detail in the next section.

Conceptual hierarchy

The two models discussed above are clearly complementary to each other. The tagged model (physical hierarchy) places the emphasis on preserving all of the information associated with the form of a dictionary entry; however, it does not offer a natural way of making explicit statements concerning structural relationships between its individual data fragments. On the other hand, concentrating on the lexical content of a dictionary (logical hierarchy model), while facilitating browsing and opportunistic searches by semantic properties, removes the database some distance away from the presentation aspects embodied in the source. A hybrid scheme represents an attempt to develop a representational framework capable of both supporting access by content and maintaining presentation information. A generalisation of a particular technique, designed explicitly to support an open-ended range of requests from an on-line dictionary (Alshawi *et al.*, 1989), aims at bringing the perspectives of the physical and logical hierarchies together. A *two-level* model of the dictionary retains an arbitrarily deeply tagged isomorph of the source as the primary repository of lexical data; at a separate level, a set of arbitrarily complex and interrelated indices is used to implement any statement concerning the content and/or form of the dictionary.

The term «two-level» reflects the internal organisation of the information found in a dictionary. The 'raw' type-setting tape is considered to be the primary repository of lexical data. However, since it has not been processed in any way at all, it retains all information required for recovery of the print form. In addition, arbitrarily complex analyses can be run on the source, associating segments (or sets of entry fragments) with a logical label. Rather than recording the results of these analyses in an identifiable, stand-alone structure (such as a tree representation of an entry, as in the case of logical hierarchy; or a tagged character stream, as in the case of physical hierarchy), they are stored implicitly in an index table. Such a table could represent any feature or property of dictionary entries. For instance, it might store pointers to all entries (and positions within them) with certain print characteristic, such as a segment in bold typeface embedded in an italic string. It might list entries with certain configurational properties, such as those having more than one subdefinition at least one of which contains a parenthetical expression. It might store results of any linguistic computation, such as recovery of elided and compacted information. It might even store completely new information derived on the basis of the data in the source. Moreover, there is no limit on the number or types of index tables which can be 'overlayed' above the base layer of data. Consequently, any combination of properties, regardless of whether they correspond to form or content, can be used as an access route into the on-line dictionary.

Since now it is possible to have a mixture of lexical attributes and print tags describing the dictionary source, it is necessary to impose some structure on them. This will streamline access and enable both meeting linguistic and/or typographic constraints when constructing queries, and systematically accommodating a potentially open-ended set of atributes. Following the natural organisation of such attributes — in a hierarchy— we view this dictionary model as based on a *conceptual hierarchy*.

A particular realisation of LDOCE within this framework exploits the following (lexical) attributes:

```
{ToT
   {pronunciation
      {no_sylls}
      {syllable*
         {stress} {onset} {peak} {coda}}}
   {syn
      {cat}
      {g_code*}
      {label}
      {orthography}
      {compound_field_1} {compound_field_2}}
   {sem
      {antonym} {synonym}
      {class}
      {box_code} {subj_code}
      {defn}
      {word*}
      {order}
      {x_ref*}}}
```

On the basis of this particular analysis, the query concerning the lexical hypothesis discussed earlier would be phrased as follows:

```
{query:
   {entry:
      {syn:
         {gcode: {And T1 I0}}}
      {sem:
         {word:  "express"}
         {word:  "to"}
         {order: "to" "express"}}}}
```

Glossing over details, this query specifies search by both syntactic and semantic attributes (such as specific grammar codes defining transitivity, and particular words used in the definition), further constrained by a print characteristic (such as word order).

There are several interesting aspects of this model. Note that since the notion of

conceptually structured lexical types is removed from the physical dictionary in storage, the description of the data need not be limited any more to traditional dictionary notions like 'headword', 'definition', and so forth. Furthermore, the extra level of description now allows the addition to it of a whole family (also potentially open-ended) of lexical attributes, and their values, computable from the source. The notion of **word_order** is a simple example; more interesting cases are **class** (computes a semantic type for a headword; see, for instance, Alshawi, 1989), or the cluster of attributes grouped under **pronunciation** (they impose explicit structure on the pronunciation fields in the dictionary and enable searches like «... three-syllable words whose second syllable has a *schwa* as a peak, and whose third syllable has a coda that is voiced stop...»; see Carter, 1989).

Such flexibility of representation comes at a cost: if a particular lexical property is not represented (as a pre-computed index table), a query based on it is clearly impossible to construct. This may refer even to features of the input which are 'obviously' easily, and readily, identified. However, this is no worse than the limitations of the tagged (physical hierarchy) model: a query related to phonological analysis is impossible to state over the electronic OALDCE, for instance. The crucial difference, however, is in 'updateability' of the descriptive templates. The constraints on representation striving to maintain the physical characteristics of the data make it impossible to incorporate in it concepts removed from presentation aspects. On the other hand, even if certain lexical properties are missing from a given dictionary description (notice for instance, how the conceptual template above 'ignores' the notion of an implicit cross-reference, discussed earlier), new index tables corresponding to the addition of these properties to the template can be created and incorporated, incrementally, into the second level of the dictionary model. Moreover, this can be done without reparsing the source, and without global redefinition of the template structure; as we discuss in the next section, this is an improvement over any of the hierarchical models presented so far.

3. Comparative evaluation

It is clear that each of the four models discussed here has certain advantages, as well as disadvantages. The choice of a particular representational framework depends ultimately on the functional requirements of the processing environment; we discuss this in more detail in the next section. It is important to realise, however, that these models are not equivalent —in the sense that equal functionality could be assigned to all of them simply by means of appropriately designed interface (i.e. access and retrieval) components.

There are several parameters which should be taken into account when comparing the database models presented earlier. The direction of the mapping between a dictionary and a database is important, as it makes either form, or content, of primary concern. Next, the acquisition paradigm also highlights certain requirements of the representational framework. The deterministic mode of lexicon acquisition, where the nature of the data in the dictionary and its relation to a computational lexicon is well understood, requires efficient access to specific dictionary fragments. On the other hand, the more opportunistic and exploratory studies of dictionary contents, aimed at discovering implicitly encoded lexical semantic properties and exploiting the notion

of distributed lexical knowledge, is concerned not so much with efficiency of access, but with flexibility. What is essential to this particular paradigm is the ability to specify projections, defined configurationally, into the space of highly structured dictionary entries: consequently, the emphasis here would be on logical content of the source, rather than its physical presentation aspects. Finally, there is the question of ease of update, where update is considered to be a task requiring not only addition of data, as more of a dictionary source gets analysed, but also modification of the underlying 'template' (description of the dictionary) as further analyses of the contents of the entry are carried out by various lexical computations.

From the perspective of limited access to entry segments, via a narrow bandwidth channel constrained by very specific requirements of a lexical data extraction procedure, the relational model of a dictionary offers the convenience of existing systems plus the advantages of efficiency of implementation.[3] Any of the algorithms for acquiring fragments of a computational lexicon (e.g. values for certain features including **subcat, human, ergative, genus,** and so forth) can be integrated within a standard relational database package; the work mentioned earlier by Nakamura and Nagao (1988) and Fontenelle and Vanandroye (1989) exemplifies just this approach. However, as we already discussed in that context, viewing lexical data as a fixed set of tables obscures a number of global lexical relations and properties; in particular the relational model inhibits discovery of implicitly encoded lexical semantic information.

Opportunistic browsing, as well as exploiting the fine distinctions in word characterisation promoted by current approaches in computational lexical semantics, are best supported by the hierarchical model of dictionary data. Clearly, it is the logical hierarchy that is of interest here, since the directionality of lexical processing is from a source dictionary, via an on-line dictionary database, to (fragments of) a computational lexicon. The complementary concerns of retaining the formal characteristics of a dictionary, typical of lexical processing preceding the generation of a printed entry, are met by the physical hierarchy model. We reiterate here the essential differences between the two.

Both models fit lexical intuitions. The real distinction then is that of how these intuitions fit the 'semantics' of the representation: trees encoding content and strings representing format. The emphasis on fine-grained structural analysis, typical of physical hierarchy models, greatly facilitates discovery of interesting —and long range— lexical semantic relationships. Notions like semantic fields, thematic roles, lexical taxonomies, conceptual networks, and so forth turn out to be naturally identifiable with structural patterns over dictionary entry analyses defined as tree shapes. On the other hand, configurational regularities in the source text —such as frequency of word use or commonality in definitions— are best stated in terms of constraints over strings localised at certain positions in dictionary entries.

This difference of emphasis —function vs. form— together with the ability (and need) to represent arbitrarily fine-grained analyses, accounts for the difference in expressive power, especially with respect to multi-funcionality of dictionary entry seg-

3. Relational systems are also used in computer assisted human lexicography, for large scale data management: see, for instance, Clear (1987) for a discussion on the advantages of such systems for maintaining a text corpus and a source dictionary management system during the production of the Collins COBUILD Dictionary.

ments. The logical hierarchy model accommodates this need easily; the physical hierarchy has no room for data duplication. As a consequence, the two models also differ in the nature of the data to be found at the terminal position in the respective hierarchical representations. The 'clean' and strongly typed strings, associated with functionally labelled nodes, are immediately usable by lexical search and extraction procedures; the arbitrary intermix of data fragments and physical tags makes separation of form and content 'on the fly' sufficiently intrusive, to the extent that any such process obscures regularities which are stated in terms of constraints both on shape of trees and content of terminal nodes.

In a sense, the two models make different commitments to the directionality of lexical processing. While it may be argued that the same kind of transduction machinery could (and does: see, e.g. Neff and Boguraev, 1991) derive both kinds of analyses from the same source, this does not make the two representation formats equivalent. While representing content imposes no boundaries on granularity of data and allows for arbitrarily deep rendering of lexical detail, representing form is constrained by the requirement that eventually a 'tree walk' should yield a print form. Consequently, there are limits to the depth of analysis which the physical hierarchy model can naturally accommodate: there is no room in a tagged dictionary format to store results of processes like typing of typical arguments, decompacting grammar codes, assigning deep phonological structure to pronunciations, and so forth.

As a further consequence from the distinctions in expressive power, and in particular those stemming from the inherent differences between trees and strings, the two models typically promote functionally different query mechanisms. Such differences are best thought of in terms of comparing tree- with string calculus; typical examples are, for instance, the Lexical Query Language (LQL) developed by Byrd (1989) and the PAT system developed for accessing tagged dictionary sources (Gonnet, 1987; Raymond and Blake, 1987). It is the natural affinity between trees and paths in a query mechanism like LQL which promotes access by function and enables semantically interesting queries to be stated in purely structural terms. Similarly, searching through semi-infinite character strings on the basis of tag labels gives PAT the tremendous power as a general string processing tool, capable, for instance, of producing concordances and frequency counts, in context and from a wide range of different perspectives.

While attempting to bring together the representational characteristics of the logical and physical hierarchical models, the two-level dictionary representation also suffers from certain drawbacks. On the good side of the conceptual hierarchy model is the natural line of integration between form and content. Equally attractive is its flexibility in both descriptive and explanatory power: the dictionary template can be updated and refined to an arbitrary level of detail, accommodating information several times removed from the source. Note that this detail need not only relate to structural notions: lexical computations can derive explicit word properties, as well as generate auxiliary resources, such as concordance and frequency data. The open-endedness of the representation makes it possible still to explore a wide range of lexical hypothesis. The separate level of interleaved index tables not only enables this open-endedness; it also makes the basis for efficient access, thus making the model a realistic alternative not only in an opportunistic, but also in a deterministic mode of dictionary use.

In principle, the content-derived notion of distributed lexical information and

the methods for associating structural patterns with long range semantic information is applicable to the conceptual hierarchy model in the same way in which it was defined in the logical hierarchy framework. In practice, however, while quite capable of supporting opportunistic browsing, the kind of on-line lexical experiments which discover lexical semantic relations as encoded via configurational regularities across the entirety of dictionary sources are best performed on pure logical representations (Boguraev, 1991).

The key point about the conceptual hierarchy model, then, is its extensibility. What the model loses in terms of specific realisations of dictionaries not having a complete outline of all lexical information, it gains in ease of update. The only requirement for upgrading an existing lexical database with more information extracted or derived from its dictionary is that a local parsing procedure be developed, capable of identifying the relevant entry segments and running the extraction algorithm over them. There is no need to reparse the whole dictionary, nor is it necessary to reorganise significantly its description (template). Finally, the model allows for incremental elaboration of dictionary description, as the template grows; particularly relevant to this is the fact that, due to the open-ended nature of the set of lexical conceptual tags, conflicts are avoided between information already existing in the database and that newly derived from the dictionary.

4. Conclusion

In the discussion so far, we have somewhat (deliberately) mixed the terms «dictionary» and «lexicon». This has been justified by the organisational principles underlying existing computational lexicons. In particular, regarding such an object as a list of words with suitably structured sets of feature-value pairs associated with them makes it possible to assume the same representational framework used both for the on-line dictionary from which lexical information is being extracted and for the computational lexicon. So far, this ambiguity in the interpretation of the term while discussing computational models has made it possible to highlight the view that all of the representations sketched above are designed as 'place holders' for existing MRD sources. As such, they implicitly start from a very general set of principles of dictionary organisation, which have evolved as a consequence of the form (natural language) and medium (sequential books) used in present day dictionaries. However, ultimately the goal of computational lexicography and lexicology is the design and construction of new kinds of dictionaries —either by using computers, e.g. for representing the complex nature of lexical data and relationships, or for use by computers, e.g. for natural language processing. The goals are only achieved by having access to a lexical knowledge base, or a «computational lexicon», which facilitates such representation and use. This alternative interpretation of the term must not assume that whatever framework emerges as a computational model of the lexicon, in the former sense, could (or should) be used for the latter sense.

Indeed, there is a growing realisation that «the traditional dictionary entry is trying to do what the language simply will not allow to be done» (see e.g. Atkins, 1991, on the inadequacy of linear sense definition, subordinated by hierarchically organised sense distinctions, in conveying the rich interdependencies between words and word senses). It follows, then, that the practice of lexicography should be re-eva-

luated, not to suit the computer, but because of it. More specifically, novel ways of structuring lexical knowledge on a large scale need to be developed, which will necessarily depart from any of the inherently impoverished models presented earlier. Some considerations of impact to lexical knowledge base design, on the basis of more elaborate studies of lexical knowledge and linguistic generalisations that need to be represented in such a knowledge base, are presented in Pustejovsky and Boguraev (1991) and Boguraev and Levin (1990).

We conclude here by reiterating our position that the computational frameworks for dictionary representation discussed earlier are not functionally equivalent, and that choice of any particular one should be influenced primarily by the nature of the lexical processing task that an on-line version of a printed dictionary is intended to support. Ultimately, a lexical knowledge base should be designed on the basis of recent developments in the area of knowledge representation, which specifically address the issue of designing a framework for compact, efficient, and non-linear encoding of lexical properties. However, the process of populating such a knowledge base with lexical knowledge extracted from dictionaries critically relies on suitable dictionary representation models.

References

ALSHAWI, H., BOGURAEV, B., and CARTER, D. (1989) «Placing LDOCE on-line», in Boguraev, B. and Briscoe, E. (eds.) *Computational Lexicography for Natural Language Processing*, Longman, Harlow and London, 41-64.

ALSHAWI, H. (1989), «Analysing the dictionary definitions», in Boguraev, B. and Briscoe, E. (eds.) *Computational Lexicography for Natural Language Processing*, Longman, Harlow and London, 153-169.

AMSLER, R. and TOMPA, F. (1988) «An SGML-based standard for English monolingual dictionaries», *Proceedings of the 4th Annual Conference of the UW Center for the New OED*, Waterloo, 61-79.

ATKINS, B. (1991) «Building a lexicon: beware of the dictionary», to appear in *Challenges of Natural Language Processing*, Bates, L. and Weischedel, R. (eds.). Cambridge University Press, 1991.

BOGURAEV, B. (1991) «Building a lexicon: the contribution of computational lexicology», to appear in *Challenges of Natural Language Processing*, Bates, L. and Weischedel, R. (eds.), Cambridge University Press.

BOGURAEV, B. and LEVIN, B. (1990) «Models for lexical knowledge bases», *Proceedings of the 6th Annual Conference of the UW Center for the New OED*, Waterloo, 65-78.

BYRD, R. J.(1989) «LQL user notes: an informal guide to the lexical query language», Research Report RC 14853, IBM Research Center, Yorktown Heights, New York.

CARTER, D. (1989) «LDOCE and speech recognition», in Boguraev, B. and Briscoe, E. (eds.) *Computational Lexicography for Natural Language Processing*, Longman, Harlow and London, 135-152.

CLEAR, J. (1987) «Computing», in Sinclair, J. (ed.) *Looking Up: An Account of the* COBUILD *project in Lexical Computing*, Collins ELT, London and Glasgow, 41-61

FONTENELLE, T. and VANANDROYE, J. (1989) *Retrieving ergative verbs from a lexical database*, MS, English Department, University of Liege.

GONNET, G. (1987) «Examples of PAT», Technical Report OED-87-02, University of Waterloo Center for the New Oxford English Dictionary, Waterloo, Ontario.

KAZMAN, R. (1986) *«Structuring the text of the Oxford English Dictionary through finite state transduction»*, University of Waterloo Technical Report No. TR-86-20.

LEVIN, B. (forthcoming) «The representation of semantic information in the lexicon», in Walker, D., Zampolli, A. and Calzolari, N. (eds.) *Automating the Lexicon: Theory and Practice in a Multilingual Environment*, Oxford University Press, Oxford, UK.

NAKAMURA, J. and NAGAO, M. (1988) «Extraction of semantic information from an ordinary English dictionary and its evaluation», *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary, 459-464.

NEFF, M., BYRD, R. and RIZK, O. (1987) «Creating and querying hierarchical lexical data bases», *Proceedings of the 2nd ACL Conference on Applied Natural Language Processing*, Austin, TX, 84-93.

NEFF, M. and BOGURAEV, B. (1989) «Dictionaries, dictionary grammars and dictionary entry parsing», *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 91-101.

NEFF, M. and BOGURAEV, B. (1991) «From machine-readable dictionaries to lexical data bases», *International Journal of Lexicography* (forthcoming).

Oxford Advanced Learner's Dictionary: Electronic (1988), Oxford University Press, Oxford, UK.

PROCTER, P. (1978) «Longman Dictionary of Contemporary English», Longman, Harlow.

PUSTEJOVSKY, J. and BOGURAEV, B. (1991) «Lexical knowledge representation and natural language processing», IBM Journal of Research and Development, vol. 35 (4).

RAYMOND, D. and BLAKE, E. (1987) «Solving queries in a grammar-defined OED», Unpublished Technical Report, University of Waterloo Center for the New Oxford English Dictionary, Waterloo, Ontario.

SPERBERG-MCQUEEN, M. and BURNARD, L. (1990) «Guidelines for the encoding and interchange of machine-readable texts», ACH, ACL, and ALLC: Draft Version 0.